



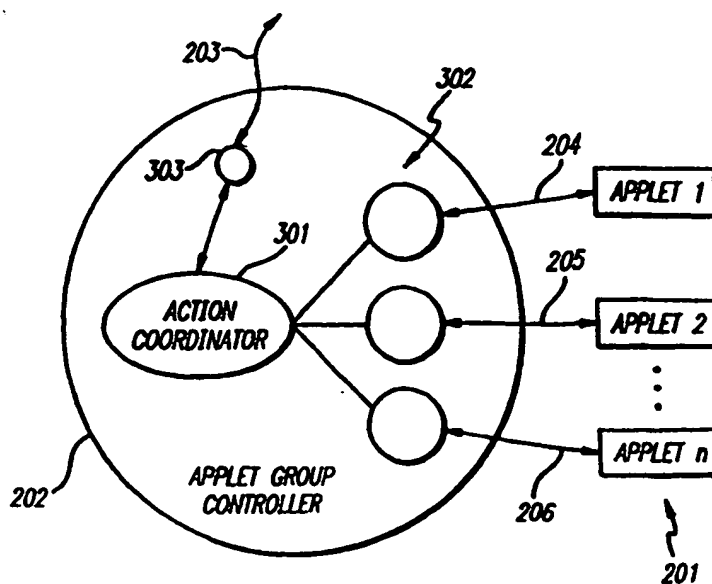
## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification <sup>6</sup> : <b>H04L 29/06</b>	<b>A1</b>	(11) International Publication Number: <b>WO 98/44695</b> (43) International Publication Date: <b>8 October 1998 (08.10.98)</b>
(21) International Application Number: <b>PCT/US98/06227</b> (22) International Filing Date: <b>30 March 1998 (30.03.98)</b> (30) Priority Data: 08/825,538                      31 March 1997 (31.03.97)                      US (71) Applicant: <b>APPLE COMPUTER, INC. [US/US]; 1 Infinite Loop, M/S: 38-PAT, Cupertino, CA 95014 (US).</b> (72) Inventors: <b>FERRIS, Michael; 434 Sunny Slope, Oakland, CA 94610 (US). POPP, Nicolas; 196 Santa Monica Avenue, Menlo Park, CA 94025 (US). FORSTALL, Scott; 2349 Greer Road, Palo Alto, CA 94303 (US). D'HARCOURT, Charles; 571 Haight Street, San Francisco, CA 94117 (US).</b> (74) Agents: <b>HECKER, Gary, A. et al.; Hecker &amp; Harriman, Suite 1600, 2029 Century Park East, Los Angeles, CA 90067 (US).</b>		(81) Designated States: <b>CA, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).</b> Published <i>With international search report.</i> <i>Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>

(54) Title: **METHOD AND APPARATUS FOR UPDATING AND SYNCHRONIZING INFORMATION BETWEEN A CLIENT AND A SERVER**

## (57) Abstract

The present invention provides a method and apparatus for integrating applets running on a client with the application logic for applications running on a server. The mechanism of the present invention provides two significant advantages over the prior art: the synchronization of state, and the recognition of user actions in the browser including the invocation of the appropriate application logic in the server. Instead of using the FORM element available in HTML, individual active applets (201) are available and may be created that obtain user input (e.g. a checkbox, textbox, button, etc.). A hidden applet called the Applet Group Controller (202) is created which handles communication between the applets on the browser and the application logic on the server. The Applet Group Controller (202) maintains an association (302) with each applet (201) that maintains the keys and values of any parameters and variables for the applet (201). In a preferred embodiment, upon the invocation of an event, the association instructs an Action Coordinator (301) to invoke an action. The Action Coordinator (301) obtains and transmits a list of all of the keys and their values to the server. The server invokes the appropriate application logic and transmits the keys and their values (as updated during the execution of the application logic) back to the Action Coordinator (301). The Action Coordinator (301) then pushes the updated values out to the applets (201) (through their Associations (302)) at which time the browser's display is updated with the new values.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LJ	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

## METHOD AND APPARATUS FOR UPDATING AND SYNCHRONIZING INFORMATION BETWEEN A CLIENT AND A SERVER

### BACKGROUND OF THE INVENTION

5

#### 1. FIELD OF THE INVENTION

This invention relates to the updating and synchronization of information between a client and a server. More specifically, this invention relates to the updating and synchronization of information between clients and servers on the world wide web through the use of applets.

#### 2. BACKGROUND

The Internet is a client/server system. A "client" is the computer that you use to access the Internet. When you log onto the World Wide Web portion of the Internet, you view "web pages" that are stored on a remote "server" computer. Sometimes it is possible for a client to make a change or enter data on a web page. A problem with the Internet is that the whole page needs to be resent from the server to the client with the changed or new data. This is slow and cumbersome. Some background on the Internet helps provide an understanding of this problem.

The Internet is a worldwide network of interconnected computers. An Internet client accesses a computer on the network via an Internet provider. An Internet provider is an organization that provides a client (e.g., an individual or other organization) with access to the Internet (via analog telephone line or Integrated Services Digital Network line, for example). A client can, for example, download a file from or send an electronic mail message to another computer/client using the Internet. An Intranet is an

internal corporate or organizational network that uses many of the same communications protocols as the Internet. The terms Internet, World Wide Web (WWW), and Web as used herein includes the Intranet as well as the Internet.

- 5       The components of the WWW include browser software, network links, and servers. The browser software, or browser, is a user-friendly interface (i.e., front-end) that simplifies access to the Internet. A browser allows a client to communicate a request without having to learn a complicated command syntax, for example. A browser typically provides a
- 10 graphical user interface (GUI) for displaying information and receiving input. Examples of browsers currently available include Netscape Navigator and Internet Explorer.

- A browser displays information to a client or user as pages or documents. A language called Hypertext Markup Language (HTML) is used
- 15 to define the format for a page to be displayed in the browser. A Web page is transmitted to a client as an HTML document. The browser executing at the client parses the document and produces and displays a Web Page based on the information in the HTML document. Consequently, the HTML document defines the Web Page that is rendered at runtime on the browser.

- 20       HTML is a structural language that is comprised of HTML elements that are nested within each other. An HTML document is a text file in which certain strings of characters, called tags, mark regions of the document and assign special meaning to them. These regions are called HTML elements. Each element has a name, or tag. An element can have attributes that specify
- 25 properties of the element. Blocks or components include unordered list, text boxes, check boxes, and radio buttons, for example. Each block has properties

such as name, type, and value. The following provides an example of the structure of an HTML document:

```
5.      <HTML>
        <HEAD>
        .... element(s) valid in the document head
        </HEAD>
        <BODY>
        .... element(s) valid in the document body
10     </BODY>
        </HTML>
```

Each HTML element is delimited by the pair of characters "<" and ">". The name of the HTML element is contained within the delimiting characters. The combination of the name and delimiting characters is referred to as a marker, or tag. Each element is identified by its marker. In most cases, each element has a start and ending marker. The ending marker is identified by the inclusion of another character, "/" that follows the "<" character.

HTML is a hierarchical language. With the exception of the HTML element, all other elements are contained within another element. The HTML element encompasses the entire document. It identifies the enclosed text as an HTML document. The HEAD element is contained within the HTML element and includes information about the HTML document. The BODY element is contained within the HTML. The BODY element contains all of the text and other information to be displayed.

25 Information servers maintain the information on the WWW and are capable of processing a client request. HyperText Transfer Protocol (HTTP) is the standard protocol for communication with an information server on the WWW. HTTP has communication methods that allow clients to request data from a server and send information to the server.

To submit a request, the client contacts the HTTP server and transmits the request to the HTTP server. The request contains the communication method requested for the transaction (e.g., GET an object from the server or POST data to an object on the server). The HTTP server responds to the client  
5 by sending a status of the request and the requested information. The connection is then terminated between the client and the HTTP server.

A client request therefore, consists of establishing a connection between the client and the HTTP server, performing the request, and terminating the connection. The HTTP server does not retain any  
10 information about the request after the connection has been terminated. HTTP is, therefore, a stateless protocol. That is, a client can make several requests of an HTTP server, but each individual request is treated independent of any other request. The server has no recollection of any previous request. The server does not retain state from a prior request.

15 An addressing scheme is employed to identify Internet resources (e.g., HTTP server, file or program). This addressing scheme is called Uniform Resource Locator (URL). A URL contains the protocol to use when accessing the server (e.g., HTTP), the Internet domain name of the site on which the server is running, the port number of the server, and the location of the  
20 resource in the file structure of the server.

The WWW uses a concept known as hypertext. Hypertext provides the ability to create links within a document to move directly to other information. To activate the link, it is only necessary to click on the hypertext link (e.g., a word or phrase). The hypertext link can be to information stored  
25 on a different site than the one that supplied the current information. A URL is associated with the link to identify the location of the additional

information. When the link is activated, the client's browser uses the link to access the data at the site specified in the URL.

If the client request is for a file, the HTTP server locates the file and sends it to the client. An HTTP server also has the ability to delegate work to  
5 Common Gateway Interface (CGI) programs. The CGI specification defines the mechanisms by which HTTP servers communicate with gateway programs. A gateway program is referenced using a URL. The HTTP server activates the program specified in the URL and uses CGI mechanisms to pass program data sent by the client to the gateway program. Data is passed from  
10 the server to the gateway program via command-line arguments, standard input, or environment variables. The gateway program processes the data, generates an HTML document, and returns the HTML document as its response to the server using CGI (via standard input, for example). The server forwards the HTML document to the client using the HTTP. In this  
15 respect, although the gateway programs have the ability to generate HTML, the document displayed by the browser remains a static document. Thus, each time information is requested from the server, a completely new HTML document containing the information is created, transmitted to the client, and loaded/rendered on the client's browser.

20 The WWW is often used to exchange information between the client (browser) and the server. In many circumstances, the server maintains a database and the client may desire to search, update or edit the database. In this respect, information is managed on the browser that the client is editing. Due to the fact that the client and server are often maintained on separate  
25 and distinct computer systems, changes made on the browser need to be reflected on the server and if information on the server changes, the browser

needs to reflect those changes; in other words, the information between the client and server needs to be synchronized.

HTML includes a FORM element that provides the ability to create a fill-in form for the client. An HTML document containing a FORM element permits the user to enter information. The user enters the information through the use of a limited number of FORM components: checkboxes; radio boxes; pull-down lists; text windows; and menus. Additional components cannot be created using an HTML FORM element. The information entered through the FORM components is transmitted as a single unit to a gateway program on the HTTP server. In this manner, FORMs can be used to input information. However, FORM elements may not be placed inside a HEADING element, inside another FORM, or inside character emphasis markups (i.e. STRONG or EM). As a result, all information input into a FORM must be transmitted in one unit. Thus, several FORMs must be created in order to transfer information to several different gateway programs. In this manner, if the gateway program merely requires one parameter, an entire FORM must be created to accept the parameter from the user. In addition, the FORM element does not provide functionality for synchronizing data between a client and a server.

The information collected on a form is sent to a gateway program in NAME/VALUE pairs, where NAME is the value assigned to the name attribute of the element, and VALUE is the value assigned by the user's input. For example, if the user named a pull-down list "CITY" and selected the city of "HOUSTON" from the pull-down list, the NAME/VALUE pair sent to the gateway program is "CITY/HOUSTON". In this manner, all of the information collected from the user is transmitted to the gateway program. However, the FORM element does not provide the ability to transmit the



name-value pairs from the server back to the client. Additionally, the FORM element does not identify the action to be taken by the server or the gateway program.

As described above, the gateway program (which has to be specifically  
5 designed to accept the name/value pairs from the client) performs the desired actions or methods and generates an HTML document that is transmitted back to the client (with the results of the desired action, if any). Since the server is stateless, to preserve any of the name/value pairs originally transmitted to the server, the server or CGI program must retransmit the  
10 name/value pairs back to the client in an `<INPUT TYPE = "hidden">` element, within the returned HTML document (which may contain a new FORM), as a record of the state of the client-server transaction. Therefore, each time information needs to be updated on the client's browser due to changes made during execution of the gateway program, an HTML definition  
15 that defines the Web page (including any name/value pairs within a new FORM) must be transmitted to the client for rendering on the client's display. The transmission and rendering of an entire page is time consuming and undesirable, especially when only a small portion of the page has changed. As a result, information transmitted through using the current capabilities is  
20 not efficient and cannot be displayed quickly on the client and/or server.

### SUMMARY OF THE INVENTION

The present invention provides a method and apparatus for integrating applets running on a client with applications running on a server (the client and server can be on the same or different computer systems). The mechanism of the present invention provides significant advantages over the prior art including the synchronization of information between a client and server, the recognition of user events in the browser, and the invocation of appropriate action logic in a server.

The prior art provides for the use of HTML (a programming language used to define a Web page on the Internet). The Web page definition (written in HTML) consists of numerous elements that are combined to produce the entire page. One such element provides for a fill-in form (a HTML FORM element) that allows a user (client) to input information onto a form displayed by the client. The FORM element has many limitations that the present invention addresses using individual applications or applets that run on a client. The applets are defined outside of a Web page and can be programmed to produce an unlimited number of input mechanisms to a user (e.g., checkboxes, textboxes, buttons, etc.).

In the present invention, a hidden applet called the Applet Group Controller is created which handles the communication between the applets and the server. In one embodiment, upon the invocation of an event in the browser (e.g., the pressing of a button created by an applet), an "Association" (located in the Applet Group Controller) instructs an Action Coordinator (located in the Applet Group Controller) to invoke an action (e.g., search a database on the server containing user specified values). In response, the

Action Coordinator obtains and transmits a list of all of the values received from the user, and the action to be invoked to the server.

The server invokes the appropriate functions utilizing the values transmitted from the Action Coordinator and retransmits the values

5 (including any values that may have changed as a result of the invoked action) back to the Action Coordinator. The Action Coordinator then pushes the updated values out to the applets (through their Associations). The applets update their displays in the browser's display using the new values. In this manner, the information between the client and the server is

10 synchronized quickly without the need to display a new HTML document.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 provides an example of a general purpose computer to be used with an embodiment of the invention.

Figure 2 demonstrates the relationship between applets, the Applet  
5 Group Controller, and the server according to an embodiment of the invention.

Figure 3 demonstrates an enlarged view of the Applet Group  
Controller of Figure 2.

Figure 4 demonstrates components of an Association according to an  
10 embodiment of the invention.

Figure 5 demonstrates an initialization process for the Applet Group  
Controller according to an embodiment of the invention.

Figure 6A demonstrates a client-side synchronization process according  
to an embodiment of the invention.

15 Figure 6B demonstrates a server-side synchronization process  
according to an embodiment of the invention.

Figure 7 demonstrates the relationship between Applets, their  
Associations, and the Server according to an embodiment of the invention.

### DETAILED DESCRIPTION OF THE INVENTION

A method and apparatus for synchronizing information between a client and a server is described. In the following description, numerous specific details are set forth in order to provide a more thorough description of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known features have not been described in detail so as not to obscure the invention.

The present invention can be implemented on a general purpose computer such as illustrated in Figure 1. A keyboard 110 and mouse 111 are coupled to a bi-directional system bus 118. The keyboard and mouse are for introducing user input to the computer system and communicating that user input to CPU 113. The computer system of Figure 1 also includes a video memory 114, main memory 115 and mass storage 112, all coupled to bi-directional system bus 118 along with keyboard 110, mouse 111 and CPU 113. The mass storage 112 may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology. Bus 118 may contain, for example, 32 address lines for addressing video memory 114 or main memory 115. The system bus 118 also includes, for example, a 32-bit DATA bus for transferring DATA between and among the components, such as CPU 113, main memory 115, video memory 114 and mass storage 112. Alternatively, multiplex DATA/address lines may be used instead of separate DATA and address lines.

In the preferred embodiment of this invention, the CPU 113 is a 32-bit microprocessor manufactured by Motorola, such as the 680X0 processor, 60X RISC processor, or a microprocessor manufactured by Intel, such as the 80X86,

or Pentium processor. However, any other suitable microprocessor or microcomputer may be utilized. Main memory 115 is comprised of dynamic random access memory (DRAM). Video memory 114 is a dual-ported video random access memory. One port of the video memory 114 is coupled to  
5 video amplifier 116. The video amplifier 116 is used to drive the cathode ray tube (CRT) raster monitor 117. Video amplifier 116 is well known in the art and may be implemented by any suitable means. This circuitry converts pixel DATA stored in video memory 114 to a raster signal suitable for use by monitor 117. Monitor 117 is a type of monitor suitable for displaying graphic  
10 images.

The computer system described above is for purposes of example only. The present invention may be implemented in any type of computer system or programming or processing environment. When a general purpose computer system such as the one described executes the processes and process  
15 flows described herein, it is a means for synchronizing information between a client and a server.

In the computer industry, a program that commences operation from a WWW document (an HTML document) is referred to as an "applet". An applet can be used to transmit data to a server and receive data from the  
20 server without having to transmit the entire Web page.

The present invention provides a method for synchronizing information between a client browser and a server (the client and server can be the same or different computer systems). To avoid the limitations present with the FORM element, the preferred embodiment of the present invention  
25 uses individual applets that retrieve user input and an applet controller that manages communication between the applets and the browser and server.

The applets can be written using a programming language such as Java. The advantages of a programming language such as Java include: the ability to write an unlimited number and variety of programs, Java applications will run on almost any supporting platform without the need to recompile the code, Java is widely used in WWW applications, and a Java program may be embedded into a Web page (HTML document). Further, a Java program may execute its action logic on either a client or a server (although applets often execute logic on a client's computer).

To commence an applet from an HTML document, the applet is treated as though it were an HTML element with an applet tag. The applet tag specifies the location of the class file (a class file contains a description of a specific kind of programmed object, including the instructions that are particular to that object) and the display area allocated to the applet. For example, to launch an applet that displays a calendar, the lines of HTML code might read:

```
<APPLET CODE=next.wof.widgets.CalendarApplet.class  
        WIDTH=203 HEIGHT=200>  
</APPLET>
```

20

When the browser executes the above lines of code, it will ask the computer system to load "next.wof.widgets.CalendarApplet.class". The browser also informs the computer system that the applet may draw or occupy a region that is 203 by 200 pixels (the top left corner of the applet's display area is determined by the browser based on the Web page's HTML definition). As in other programming languages, the applets may utilize various parameters that may be initialized and passed to the applet through the HTML document (as Param tags).

The HTML elements that define a Web page can be included in a single file. In a preferred embodiment of the present invention, a file that contains HTML elements is referred to as an HTML template. An HTML template may define all or a subset of the Web page. Two or more HTML templates  
5 that define portions of the Web page can be combined to provide a complete definition. An HTML template is used to generate an HTML document (or definition) for a Web page. In the preferred embodiment, an HTML template contains HTML element definitions.

In a preferred embodiment of the present invention, special HTML  
10 tags or HTML elements (including applets) are mapped to objects in an object-oriented environment. Classes of objects are defined (in a declarations file) for each HTML element as well as the HTML template (or page). By providing a one-to-one mapping between each HTML element and object classes, HTML documents can be manipulated programmatically. The  
15 properties of each element are stored in instance variables of the associated object. Each object class can include methods to manipulate the HTML element within an HTML document. Classes of objects are also defined that do not necessarily map to a particular HTML element. These object classes can generate HTML statements or manage the acquisition of data used to  
20 generate HTML, or that provide conditional functionality that varies the characteristics of an HTML definition, for example. At runtime, instances of the object classes are instantiated for the HTML elements defined in the HTML document. The following provides an example of an HTML template:



```

5  <html>
    <head>
        <title>Capitalize String</title>
    </head>
    <body bgcolor="#FFFFFF">
        <h2>Mission Critical Client-Server String Capitalization</h2>
        <hr>
        <br>
        Enter a string in the field and press the Do It button:
        <br>
        <WEBOBJECT name=INPUTFIELD></WEBOBJECT>
        <br>
        <WEBOBJECT name=BUTTON></WEBOBJECT>
        <br>
        Result:
        <br>
        <WEBOBJECT name=OUTPUTFIELD></WEBOBJECT>
        <br>
    </body>
</html>
20

```

### Table One

The above HTML template includes tags for HTML, HEAD, BODY, and WEBOBJECT elements. At runtime, the tags are used to create an object class instance (or instances). The HTML template can contain tags (e.g., TITLE, H2, HR, and BR) that define static HTML for which there is no need to instantiate an object class instance. A class is defined for each HTML element. Attribute information associated with an HTML tag can be used to initialize instance variables of an object. Methods defined for an object can generate HTML statements in an HTML document that define the HTML element associated with the object's class, for example.

The WEBOBJECT tags in Table One (INPUTFIELD, BUTTON, and OUTPUTFIELD) refer (or provide a pointer) to entries in a declarations file that map the entries to object classes and provide declarations for the tag.

35 These declarations can be used to initialize instance variables of an object and provide runtime environment information such as object, method and

variable bindings. Table Two provides an example of the contents of a declarations file that accompanies the HTML template of Table One.

```

5 INPUTFIELD :      WOAApplet {
      code = "next.webobjects.widgets.TextFieldApplet.class";
      codebase = "/Java_packages";
      width = "200";
      height = "20";
10      associationClass = "next.webobjects.widgets.SimpleAssociation";
      stringValue = inputString
    };

15 BUTTON      :      WOAApplet {
      code = "next.webobjects.widgets.ButtonApplet.class";
      codebase = "/Java_packages";
      width = "200";
      height = "20";
      associationClass = "next.webobjects.widgets.SimpleAssociation";
20      title = "DoIt";
      action = "capitalizeString"
    };

25 OUTPUTFIELD      :      WOAApplet {
      code = "next.webobjects.widgets.TextFieldApplet.class";
      codebase = "/Java_packages";
      width = "200";
      height = "20";
      associationClass = "next.webobjects.widgets.SimpleAssociation";
30      stringValue = outputString;
    };

```

Table Two

The name attribute of the WEBOBJECT tag binds a WEBOBJECT HTML template entry to the declarations file. For example, the name having a value of "INPUTFIELD" (from the WEBOBJECT tag of Table 1) binds its entry to the

INPUTFIELD entry in the declarations file. A similar approach can be taken for the BUTTON and OUTPUTFIELD values associated with the remaining WEBOBJECT tags. The INPUTFIELD, BUTTON, and OUTPUTFIELD declarations bind the WEBOBJECT tag to instances of the WOAApplet class. The WOAApplet class permits the specification of applet-specific parameters, such as the dimensions of the applet and the location of the ".class" file to download to the browser. It also allows you to initialize parameters to be

downloaded to the applet and to bind an applet's keys to variables and methods in the server. For example, the INPUTFIELD declares a dynamic element called INPUTFIELD that generates the HTML needed to download the Java applet defined in the "next.webobjects.widgets.TextFieldApplet.class" to the web browser. Further, the dynamic elements called BUTTON and OUTPUTFIELD are defined to generate the HTML needed to download the Java applet defined in "next.webobjects.widgets.ButtonApplet.class" and "next.webobjects.widgets.TextFieldApplet.class" to the web browser. Thus, the "code =" line in the declarations file maintains the location of the Java applet for each object. Further, the "width =" and "height =" lines provide the display area to be allocated for the applet. In this manner, the declaration file provides the ability to generate the HTML code for the applet (e.g. `<APPLET CODE=next.webobjects.widgets.TextFieldApplet.class WIDTH=200 HEIGHT=20></APPLET>`).

The declarations file also provides for various keys and values to which the keys are bound. For example, the INPUTFIELD instance of WOApplet has an associationClass key and a stringValue key. The associationClass key is bound to the SimpleAssociation class (Associations are discussed in detail below). The stringValue key is bound to the value of inputString. Similarly, the BUTTON instance of WOApplet has an associationClass key (bound to the SimpleAssociation class), a title key (bound to the character string "DoIt"), and an action key (bound to the character string "capitalizeString"). The values to which the keys are bound are set in a script file (discussed below). The bindings in combination with the Association class (discussed below) provide for the association of state in the applet (on the client) with state in the server and events in the applet with

the invocation of methods in the server. Consequently, information between the client and the server may be synchronized.

A script is a type of program that consists of a set of instructions that is read and interpreted by another application (e.g., an interpreter). An instruction is an expression that conforms to the rules and syntax expected by a script interpreter. A script language (set of script instructions) may include simple control statements such as loops and if/then expressions.

Since the language does not include complex constructs, the language is easy to master. As with other interpretive languages, a developer does not have to perform steps to translate a script into machine-readable code. Even though a script is interpreted at runtime, it is not considered to cause any degradation in the overall response time (the actual network response times are the primary concern). Therefore scripts are typically used to provide the logic associated with a Web page. However, the invention can be practiced using other languages such as Objective C, C++, etc. Table Three provides an example of the contents of a script file that accompanies the HTML template of Table One and the Declarations file of Table Two.

20

25

30

```
id inputString;
id outputString;

- awake {
    // Set up parameter values
    inputString = @"Scrumptious";
    outputString = @"";
}

- capitalizeString {
    outputString = [inputString uppercaseString];
    return nil;
}
```

Table Three

The above script file declares the variables `inputString` and `outputString` ("id `inputString`;" and "id `outputString`;" ) and then sets their initial values ("Scrumptious" and ""). The script file then declares the `capitalizeString` script which sets the value of the `outputString` variable to the capitalization of the value in the `inputString` variable and returns nil. The string value associated with the INPUTFIELD instance (in the declarations file of Table Two) is bound to the value of `inputString` that is set in the script file. The string value from the OUTPUTFIELD instance (in the declarations file of Table Two) is bound to the value of `outputString` that is set in the script file.

10 The action associated with the BUTTON instance (in the declarations file of Table Two) will invoke the `capitalizeString` script located in the script file.

In this manner, the HTML template, the declarations file, and the script file are all combined to define a Web page. The above created Web page provides a string capitalizer. Such an interface involves two text fields and a button. The first field is for input and the second field is for displaying the result. The user enters a value in the input field and presses the button. The output field is set to an all uppercase version of the value entered in the input field. This example further illustrates the object oriented approach and the objectifying (Objective C, C++ objects, SmallTalk objects, etc.) of HTML elements.

20

Upon execution of an applet, the applet initializes itself through an `Init` procedure that the browser invokes on the client. Subsequent to initialization, the client's browser calls the applet's `Start` routine and its `Paint` routine to draw the applet in the browser's display. Thereafter, the browser may recall the `Paint` routine to redraw/render the applet on the client's display. Upon completion of the applet, the applet's `Stop` routine is invoked by the browser.

25

Applets can display selectable items (e.g. buttons, textboxes, etc.), to elicit responses from a user, and to trigger actions. However, there is no facility for using applets to pass data between a server and an applet.

The present invention provides for a mechanism that integrates  
5 applets (e.g. Java applets) running on a browser with applications running on a server. This mechanism provides at least two significant advantages over the prior art. Firstly, the mechanism provides for the synchronization of state. Synchronization of state permits data that is present on the browser to be updated on the server (if necessary) and data that is present on the server  
10 to be updated on the browser (if necessary). Secondly, upon specified events on a browser by the user (e.g., the clicking of a button or selection of a radio button), certain actions may need to occur on the server. The present invention provides for the recognition of these events by the applets in the browser and the invocation of the actions in the server.

15 In one embodiment, synchronization of state occurs at three times. Initially, when a page is generated, all the state for which there are state bindings are sent to the client (initial set-up) (the initialization process is described in detail below). This initial synchronization ensures that the server's data is used to initialize the Web page. The other two points of  
20 synchronization surround the invocation of an action. Before an action is invoked in the server, any state that has changed in the client is transmitted back to the server (e.g., state may change when a browser user enters information in an input field of the Web page). Subsequently, after the action has completed, any state that has changed in the server is sent back to the  
25 client. In this manner, the method invoked by the action will only act on the current data (since it has just been transmitted back from the client, and afterwards, anything the action has caused to be changed, is reflected back to

the client for the user's viewing). In addition to the above, may occur at any point in time. For example, a timer may be utilized such that synchronization is to occur every five (5) seconds.

The present invention consists of numerous applets that are capable of data display and/or input. In the preferred embodiment of the present invention, various applets are provided to the user including: a TextFieldApplet (a text field permitting text input from a user); a ButtonApplet (a push button); a CheckboxApplet (a toggle button); a ChoiceApplet (a box containing non-editable items that allows multiple selection); a ListApplet (a list of items that can allow multiple selection); a RadioGroupApplet (a set of mutually exclusive buttons); a ScrollingTextApplet (a larger text entry area); CalendarApplet (a primitive graphical calendar for entering or displaying dates); and a StatusTextApplet (displays a read-only message center aligned within the applet's frame). Each applet may contain multiple values (e.g. the text in a TextFieldApplet or the selected item in the ListApplet). In addition to the applets provided, a user may program any type of applet as long as it is possible to obtain and set the value of each key that the applet and its Association will support (keys, values, and Associations are explained in detail below), and it must be possible to tell when each supported action occurs (if any) (actions are discussed in detail below). These applets act as a substitute for the FORM element of HTML documents and provide the user the ability to input/retrieve information from the user.

In addition to the above active applets, there is an additional applet, transparent to the user, that controls the other applets. This transparent applet is called the AppletGroupController ("Controller"). The Controller is responsible for managing the communication and the data synchronization

between the client and the server. In the preferred embodiment, the Controller resides on the client's computer system. Therefore, the Controller has a line of communication back to the server. Figure 2 demonstrates the relationship between applets, the Controller, and the server. An unlimited  
5 number of active applets 1 through n 201 provide for the browser's display attributes. Each applet 201 communicates with the Controller 202 which maintains a line of communication 203 with the server.

Figure 3 demonstrates an enlarged view of the AppletGroup-Controller 202. The Controller 202 includes an Association 302 to each of the  
10 applets 201. Controller 202 passes data (or state) through Associations 302 to applets 201. The Associations 302 provide a fixed interface for communications with Controller 202. As a result, Controller 202 utilizes the Associations 302 in a uniform manner. Each of Associations 302 includes an interface (204, 205, and 206) for communicating with applets 201. Different  
15 classes of Associations 302 can be used for different types of applets. Each of Association 302 has the ability to obtain and set state for its corresponding applets 201 at runtime. In addition, the Associations 302 are also responsible for knowing when to invoke the applet's supported actions. For example, the API for communication between Associations 302 and applets 201 include the  
20 ability to obtain the applet's keys, obtain the key's values, and invoke an applet's action (keys, values, and actions are discussed in detail below). Action Coordinator 301 communicates with Associations 302. Action Coordinator 301 is responsible for state synchronization and for building a package of data to be transmitted back to the server. Communication with  
25 the server occurs through Communications Channel 303.

Referring to Figure 4, Associations 302 include two types of bindings: state bindings 401 and action bindings 402. State bindings 401 are used to



associate state in the applets with state in the server and they form the basis for the state synchronization feature of the present invention. Action bindings 402 are used to associate particular events (e.g., a button being pressed in the client) with the invocation of methods in the server. Applets

5 201 trigger actions in the server using Action bindings 402.

Referring to Figure 7, each instance of applets 201 manages a set of key-value pairs 701. State bindings 401 include a list of "keys" that represent the data or state managed by applets 701. The "keys" of the association's state bindings 401 correspond to the keys in the key-value pairs 701 managed by an

10 applet 201. In addition, a key is bound to a specific object or variable 704 in the server to which it is synchronized. In a preferred embodiment, the value in a key-value pair 701 is preferably an object or an aggregation of objects that is one of four classes: string, array, dictionary, or data (referred to as a property-list type of object). Action bindings 402 consist of a mapping

15 between an event that Applet 201 triggers, for example (applets can support an unlimited number of actions), and an action on the server. An Action can be implemented, for example, as method 703. Using Action bindings 402, when an event 702 occurs, method 703 on the server is invoked.

Synchronization may occur at various points in time. In one

20 embodiment, actions provide one of the synchronization points for the state as previously discussed. State synchronization can be used to synchronize multiple keys. For example, a TextFieldApplet may have two (2) keys called "StringValue" (text) and "echoCharacter" (single character string to be used as the echo character used, for example, in a password field). The value for the

25 "StringValue" key is the text that the user has entered; and the value for the "echoCharacter" key is the single character that is echoed to the display device (e.g. "\*"). Thus, the TextFieldApplet has two key-value pairs (the

INPUTFIELD and OUTPUTFIELD elements from Table One are TextFieldApplets with only one key (stringValue) defined in the declarations file of Table 2). State bindings 401 are used to bind the key-value pairs of the TextFieldApplet with server state 704 (e.g., an object or variable resident on the server). Further, an action binding 402 of an instance of Associations 302 associated with the TextFieldApplet may specify a method 703 that occurs upon the happening of an event 702 (e.g., entry of a carriage return) associated with the text field. In addition, state synchronization can be used to synchronize multiple keys for multiple different applets.

Another example is that of a ButtonApplet. A ButtonApplet is another example wherein state synchronization is used to synchronize multiple keys. An instance of ButtonApplet may have two keys entitled "Title" and "Enable". The value of the "Title" key contains the title of the button that is displayed. The value of the "Enable" key determines whether the button is enabled or disabled (whether it may be depressed or not). For example, the BUTTON applet defined in Table Two contains one key "title". The action for a ButtonApplet is, for example, the event that occurs when the button is pressed. The action associated with the BUTTON applet of Table Two, for example, is capitalizeString defined in the script file of Table Three.

Referring to Figure 3, the API (Application Programming Interface) (e.g., interfaces 204-206) that the Associations 302 uses to communicate with applets 201 makes a determination as to what the applet's value for an Association's key is and what the new value for that key is (whether the value has changed). Consequently, the interface merely needs to have access to the keys and values of the applets. As a result, any existing applet may be used in the present invention as long as the applet is supported by an API

that permits the retrieval and setting of its keys and values. An Association that utilizes the API is used to communicate with the applet.

At runtime, an initialization procedure is preferably executed for Controller 202. During the initialization of Controller 202, the various  
5 associations are instantiated and connected to the applets, and initial values are parsed and distributed to the applets. The details of the Controller Initialization process are set forth in Figure 5.

Step 501 of the initialization procedure creates components 501 of the Controller 202 including, for example, Action Coordinator 301 (discussed  
10 below) and Communications Channel 303 (discussed below). In a preferred embodiment, the server assigns identification numbers (the identification numbers are later used to ensure all applets are accounted for) to each applet on the HTML page at step 502. At step 503, the server transmits the  
15 identification numbers, the HTML page (including initial parameter values for the applets) to the applet group controller 202. At step 504, the applets on the browser page are surveyed. Step 504 includes the creation of a list of all of the page's applets currently present on the client. In addition, the various initial parameters (set in the HTML page) are retrieved. One such initial  
20 parameter that the web page creator passes into each applet is the class of the Association. If an Association class is retrieved as a parameter, a connection to the Association is established using the class information supplied by the parameter. As a result, the Association class parameter is looked-up by name (utilizing common Java procedures, for example), and an instance of that class is created at step 505 (thereby creating an Association for that applet). To  
25 create an instance of the class, for example, the Java code might read:

```
newAssoc = (Association)assocClass.newInstance();
```

At step 506, an instance of Association 302 is connected to Action Coordinator 301. This connection is used by Association 302 to instruct Action Coordinator 301 to invoke an action, and by Action Coordinator 301 to set values into Association 302. Association 302 is then connected to an  
5 applet 201 at step 507. This connection consists of Association 302 obtaining a reference to Applet 201. At step 508, all of the initial values (e.g., passed in through parameter tags in the HTML document) are extracted. The initial values are then placed into a dictionary of values located on the client's computer. The dictionary (or snapshot of the keys and their current values)  
10 is used upon invocation of an action.

Controller 202 repeats the connection process (steps 505-508) until all of the applets 201 expected (an applet is expected if an identification number was transmitted from the server) have been processed. Thus, at step 509, a determination is made whether all of the expected applets 201 are present and  
15 connected to their associations 302 and to Action Coordinator 301. Each of Associations 302 maintain connection with Action Coordinator 301 and with its corresponding instance of applet 201.

If it is determined at step 509 that all of the expected applets 201 are present (i.e., an applet is found for every identification number transmitted  
20 by the server) and initialized, the initial values are pushed from the dictionary to the Associations 302 and to the appropriate Applets 201.

Figure 3 illustrates an initialized Applet Group Controller 202. As a result of the above initialization process, Communications Channel 303, Action Coordinator 301, and instances of Associations 302 were created; the  
25 Associations 302 are connected to Action Coordinator 301 and corresponding Applets 201; and a dictionary of the keys and their values was created. All of

the values in the dictionaries (as transmitted from the server) are transmitted to Applets 201 thereby providing initial synchronization between the client and the server.

Referring to Figures 3 and 6A, in a preferred embodiment, upon the  
5 occurrence of an event (e.g., the clicking of a button icon), an instance of Associations 302 recognizes that the event has occurred at step 601 of Figure 6A, and determines whether the event requires the invocation of an action at step 602 (e.g. the instance of Associations 302 determines whether an action associated with the clicked button should be invoked). If no action is  
10 required, processing continues at step 601. If Associations 302 determine that an action should be invoked, it instructs Action Coordinator 301 at step 603 to invoke the action.

Upon receiving invocation instructions from the instance of Association 302, Action Coordinator 301 queries Associations 302 for the  
15 values for all of their keys at step 604. The values retrieved are examined to determine whether the value has been changed since the last communication with the server (e.g., the values retrieved are compared to the values in the dictionary). If any values have changed, synchronization with the server takes place. Action Coordinator 301 constructs a package of data to be sent to  
20 the server at step 605. In the preferred embodiment of the present invention, the package includes a list of only those properties that have changed in the browser since the last communication with the server and the action that is to be invoked on the server.

Action Coordinator 301 transfers the package to Communications  
25 Channel 303. Communications Channel 303 provides a standard interface to Controller 202 and implements an arbitrary channel of communication 203

back to the server. Communications Channel 303 transmits the package to the server at step 606 via channel of communication 203. Communications Channel 203 may be implemented in a variety of different communication protocols. However, in the preferred embodiment of the present invention, Communications Channel 203 is implemented in HTTP to communicate with the server in the same manner as a browser. Further, by using HTTP, the present invention uses a communication channel that is already being passed through a firewall (a firewall is machine that fits between a local network (e.g., a company's internal network) and the internet and limits the access that can pass between the internal local network and the internet as a whole) in order to allow access to the web site. Consequently, by using HTTP, there is no need to create a new hole in the firewall.

Referring to Figure 6B, upon receipt of the package from the browser, the server determines the appropriate action logic (e.g., method, function, or script) at step 608. The server then binds the logic's variables to the keys received at step 609. For example, the server binds the variables contained in a script using the keys transmitted from the client. Using the bindings, the variables are updated with the new state to which they are bound. As a result, prior to the invocation of the action, the new values are established in the variables that are used by the action.

The server invokes the action at step 610. Subsequent to the invocation of the action, the server constructs a package 611 at step 611 to be sent back to the browser. In constructing the package, the server determines whether a new HTML document/page is to be rendered. If a new page is to be rendered, an HTML document is created and the URL for that document is transmitted back to the Applet Group Controller 202 in the browser. The browser then retrieves the HTML document from the URL. However, if the

state merely needs to be synchronized on the browser, there is no need to generate a new page. Instead, the bindings are evaluated and the states are packaged together (e.g., the values for the variables contained in the action logic are retrieved and used to create key-value pairs). In the preferred embodiment, only those values that have changed since the last communication with the browser are compiled into the package. The package is then transmitted at step 612 through channel of communication 203 to Communications Channel 303 in Controller 202. Further, in a preferred embodiment, the values that have changed are transmitted and synchronization occurs regardless of whether a new HTML page is to be rendered.

The package is transferred from Communications Channel 303 to Action Coordinator 301. In one embodiment, the values that have changed are identified at step 613 (in such an embodiment, the package contains all of the values regardless of whether they have changed). The appropriate values are then transmitted to the appropriate instances of Association 302 at step 614. The instances of Associations 302 distribute the values back to Applets 201 at step 615. Instead of rendering a whole new page (as in the prior art), the applet updates the Web page using the updates received from Associations 302 on the display device if required at step 616.

As previously discussed, in the preferred embodiment of the present invention, Action Coordinator 301 maintains a dictionary of the keys and their values at the time of the last communication with the server (referred to as a snapshot of the keys and their values). When Action Coordinator 201 queries Associations 302 to determine the keys and their values, a comparison is performed between the values contained in the dictionary and the values stored in Associations 302. Only those values that have changed

since the last communication with the server are compiled into the package that is transmitted to the server. In addition, in the preferred embodiment of the present invention, the server also maintains a dictionary containing the keys and their bindings to application logic variables (bindings) at the time of  
5 the last communication with Controller 202 (a snapshot). Prior to transmitting the package back to Controller 202, all of the bindings are evaluated and only those values that have changed during the course of execution of the action are packaged and transmitted back to the controller 202. In this manner, only those bindings that require synchronization are  
10 transmitted to the server and vice versa.

According to one embodiment of the invention, an Applet Group Controller is initialized which includes creating various components (e.g., an action coordinator, communications channel, and associations). Connections are formed between the components of Applet Group Controller 202. A  
15 snapshot is created of the current state preferably using a dictionary of keys and their values. Upon occurrence of an event, an association instructs the action coordinator to invoke an action and the keys and values are surveyed. A package of keys and their values is constructed and transmitted to the server where the values are bound to variables and action logic is invoked.  
20 Upon any change in the values (as a result of the execution of the action logic), a package of changed values and their keys is constructed and transmitted back to the Applet Group Controller. The changed values are pushed to the appropriate applets (through their Associations) and the values are displayed by the browser. In this manner, the information is updated and  
25 synchronized between the client and the server.

Thus, a method and apparatus for synchronizing information between a client and a server is provided.



CLAIMS

1. A method for synchronizing information between a client and a server comprising the steps of:
  - initializing a communication system, said communication system
  - 5 comprising a plurality of objects, said objects capable of:
    - maintaining keys and values;
    - displaying a means to accept user input; and
    - determining when to invoke an action;
  - constructing a first package;
  - 10 transmitting said first package to said server;
  - constructing a second package;
  - transmitting said second package to said client; and
  - updating said keys and said values.
- 15 2. The method of claim 1 wherein said initialization step comprises the steps of:
  - creating a controlling object;
  - surveying said plurality of objects;
  - creating associations;
  - 20 connecting said associations to said plurality of objects;
  - creating an action coordinator; and
  - connecting said associations to said action coordinator.
- 25 3. The method of claim 2 wherein said initialization step further comprises the steps of:
  - extracting said keys and said values from said plurality of objects; and
  - creating a client dictionary of said extracted keys and extracted values
  - on said client.

4. The method of claim 1 wherein said first package is comprised of:

said keys; and

5 said values.

5. The method of claim 3 wherein said step of constructing a first package comprises the steps of:

comparing said keys and said values to said dictionary keys and said  
10 dictionary values; and

placing said keys and their corresponding values that have changed in said first package.

6. The method of claim 2 wherein said step of transmitting said  
15 first package comprises the steps of:

transmitting said first package from said action coordinator to a communications channel; and

transmitting said first package from said communications channel to said server.

20

7. The method of claim 1 wherein said step of transmitting said first package further comprises the step of binding said keys to variables in said server.

25 8. The method of claim 1 wherein said step of initializing a communications system further comprises the step of creating a server dictionary of said keys and said values on said server.

9. The method of claim 8 wherein said step of transmitting said first package further comprises the steps of:

updating said server dictionary with said keys and said values from said package; and

5 binding said server dictionary's values to variables in said server.

10. The method of claim 4 wherein said step of transmitting said first package to said server further comprises the step of binding said values from said first package to variables in said server.

10

11. The method of claim 10 wherein said second package comprises updated keys and updated values.

12. The method of claim 8 wherein said step of constructing a second package further comprises the steps of:

15

comparing said bound variables with said server dictionary's values; and

placing said keys and said values that have changed in said second package.

20

13. The method of claim 6 wherein said step of transmitting said second package comprises the steps of:

transmitting said second package from said server to said communications channel; and

25 transmitting said second package from said communications channel to said action coordinator.

14. The method of claim 3 wherein said step of updating said keys and said values further comprises the step of updating said client dictionary with said keys and said values from said second package.

5 15. The method of claim 11 wherein said step of updating said keys and said values comprises the steps of:

transmitting said keys and said updated values to said objects; and  
displaying said updated values.

10 16. The method of claim 2 wherein said step of updating said keys and said values further comprises the steps of:

transmitting updated values from said second package to said action coordinator;

15 transmitting said updated values from said action coordinator to said associations;

distributing said updated values from said associations to said objects;  
and

displaying said updated values.

17. An article of manufacture comprising:

a computer usable medium having computer readable program code embodied therein for synchronizing information between a client and a server, the computer readable program code in said article of manufacture

5 comprising:

computer readable program code configured to cause a computer to initialize a communication system, said communication system comprising a plurality of objects, said objects containing computer readable program code configured to cause said computer to:

10 maintain keys and values;

display a means to accept user input; and

determine when to invoke an action;

computer readable program code configured to cause said computer to construct a first package;

15 computer readable program code configured to cause said computer to transmit said first package to said server;

computer readable program code configured to cause said computer to construct a second package;

20 computer readable program code configured to cause said computer to transmit said second package to said client; and

computer readable program code configured to cause said computer to update said keys and said values.

18. The article of manufacture of claim 17 wherein said computer readable program code configured to cause said computer to initialize a communication system further comprises:

- 5 computer readable program code configured to cause said computer to create a controlling object;
- computer readable program code configured to cause said computer to survey said plurality of objects;
- computer readable program code configured to cause said computer to create associations;
- 10 computer readable program code configured to cause said computer to connect said associations to said plurality of objects;
- computer readable program code configured to cause said computer to create an action coordinator; and
- 15 computer readable program code configured to cause said computer to connect said associations to said action coordinator.

19. The article of manufacture of claim 18 wherein said computer readable program code configured to cause said computer to initialize a communication system further comprises:

- 20 computer readable program code configured to cause said computer to extract said keys and said values from said plurality of objects; and
- computer readable program code configured to cause said computer to create a client dictionary of said extracted keys and extracted values on said client.

20. The article of manufacture of claim 17 wherein said first package is comprised of:

said keys; and

said values.

5

21. The article of manufacture of claim 19 wherein said computer readable program code configured to cause said computer to construct a first package comprises:

computer readable program code configured to cause a computer to  
10 compare said keys and said values to said dictionary keys and said dictionary values; and

computer readable program code configured to cause a computer to place said keys and their corresponding values that have changed in said first package.

15

22. The article of manufacture of claim 18 wherein said computer readable program code configured to cause said computer to transmit said first package comprises:

computer readable program code configured to cause a computer to  
20 transmit said first package from said action coordinator to a communications channel; and

computer readable program code configured to cause a computer to transmit said first package from said communications channel to said server.

25 23. The article of manufacture of claim 17 wherein said computer readable program code configured to cause said transmit said first package further comprises computer readable program code configured to cause a computer to bind said keys to variables in said server.

24. The article of manufacture of claim 17 wherein said computer readable program code configured to cause said computer to initialize a communications system further comprises computer readable program code  
5 configured to cause a computer to create a server dictionary of said keys and said values on said server.

25. The article of manufacture of claim 24 wherein said computer readable program code configured to cause said computer to transmit said  
10 first package further comprises:

computer readable program code configured to cause a computer to update said server dictionary with said keys and said values from said package; and

15 computer readable program code configured to cause a computer to bind said server dictionary's values to variables in said server.

26. The article of manufacture of claim 17 wherein said computer readable program code configured to cause said computer to transmit said first package to said server further comprises computer readable program  
20 code configured to cause a computer to bind said values from said first package to variables in said server.

27. The article of manufacture of claim 26 wherein said second package comprises updated keys and updated values.

25



28. The article of manufacture of claim 24 wherein said computer readable program code configured to cause said computer to construct a second package further comprises:

- computer readable program code configured to cause a computer to
- 5 compare said bound variables with said server dictionary's values; and
- computer readable program code configured to cause a computer to place said keys and said values that have changed in said second package.

29. The article of manufacture of claim 22 wherein said computer

10 readable program code configured to cause said computer to transmit said second package comprises:

- computer readable program code configured to cause a computer to transmit said second package from said server to said communications channel; and
- 15 computer readable program code configured to cause a computer to transmit said second package from said communications channel to said action coordinator.

30. The article of manufacture of claim 19 wherein said computer

20 readable program code configured to cause said computer to update said keys and said values further comprises computer readable program code configured to cause a computer to update said client dictionary with said keys and said values from said second package.

31. The article of manufacture of claim 27 wherein said computer readable program code configured to cause said computer to update said keys and said values comprises:

- 5 computer readable program code configured to cause a computer to transmit said keys and said updated values to said object; and
- computer readable program code configured to cause a computer to display said updated values.

32. The article of manufacture of claim 18 wherein said computer readable program code configured to cause said computer to update said keys and said values further comprises:

- 10 computer readable program code configured to cause a computer to transmit updated values from said second package to said action coordinator;
- computer readable program code configured to cause a computer to
- 15 transmit said updated values from said action coordinator to said associations;
- computer readable program code configured to cause a computer to distribute said updated values from said associations to said objects; and
- computer readable program code configured to cause a computer to
- 20 display said updated values.

33. The method of claim 1 further comprising the step of invoking a method.

25 34. The article of manufacture of claim 17 further comprising computer readable program code configured to cause a computer to invoke a method.

35. The method of claim 4 wherein said first package is further comprised of a reference to said action.

36. The article of manufacture of claim 20 wherein said first package  
5 is further comprised of a reference to said action.

37. The method of claim 5 further comprising the step of placing a reference to said action in said first package.

10 38. The article of manufacture of claim 21 further comprising computer readable program code configured to cause a computer to place a reference to said action in said first package.

1/5

FIG. 1

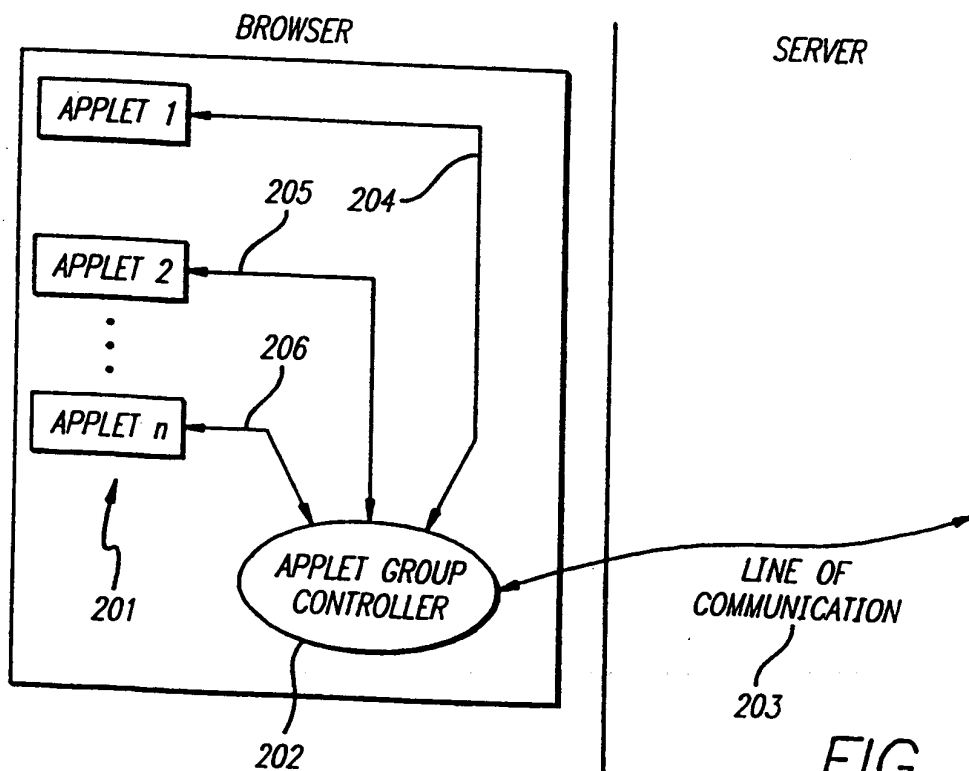
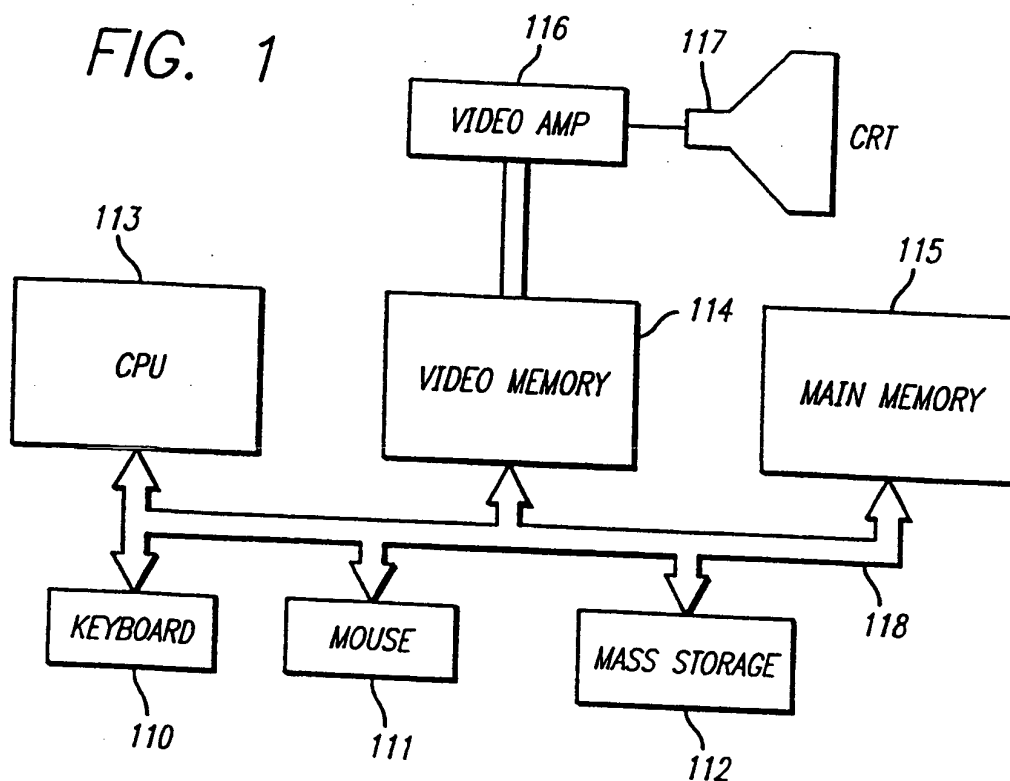
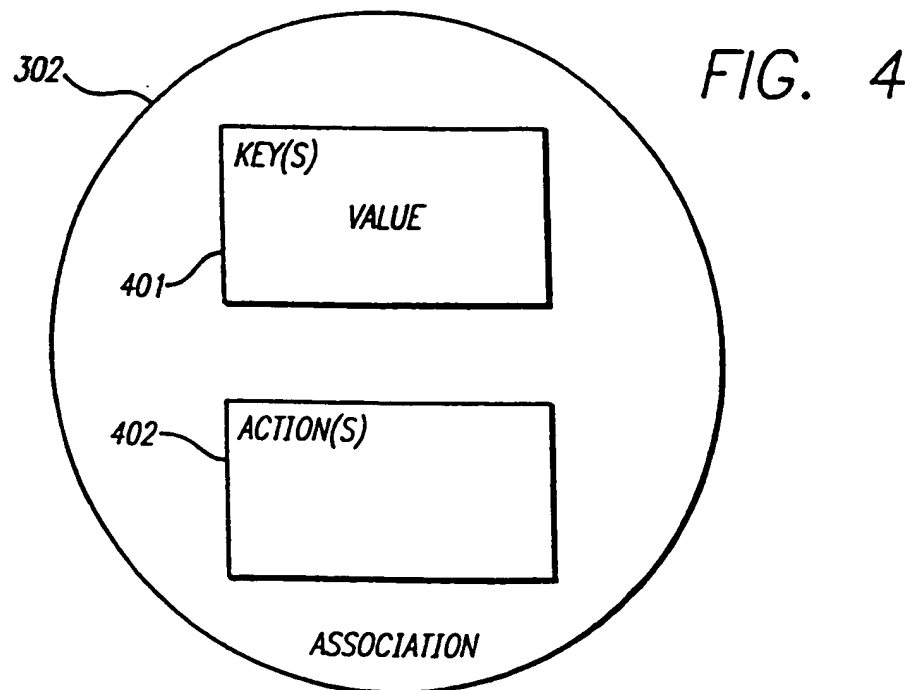
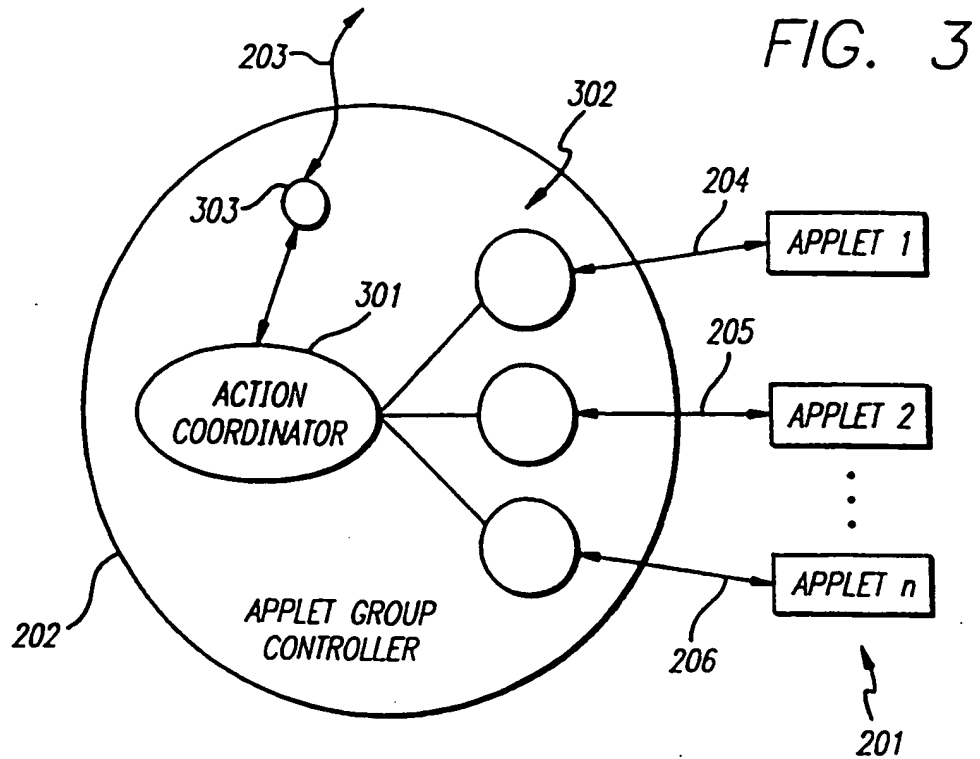


FIG. 2

2/5



3/5

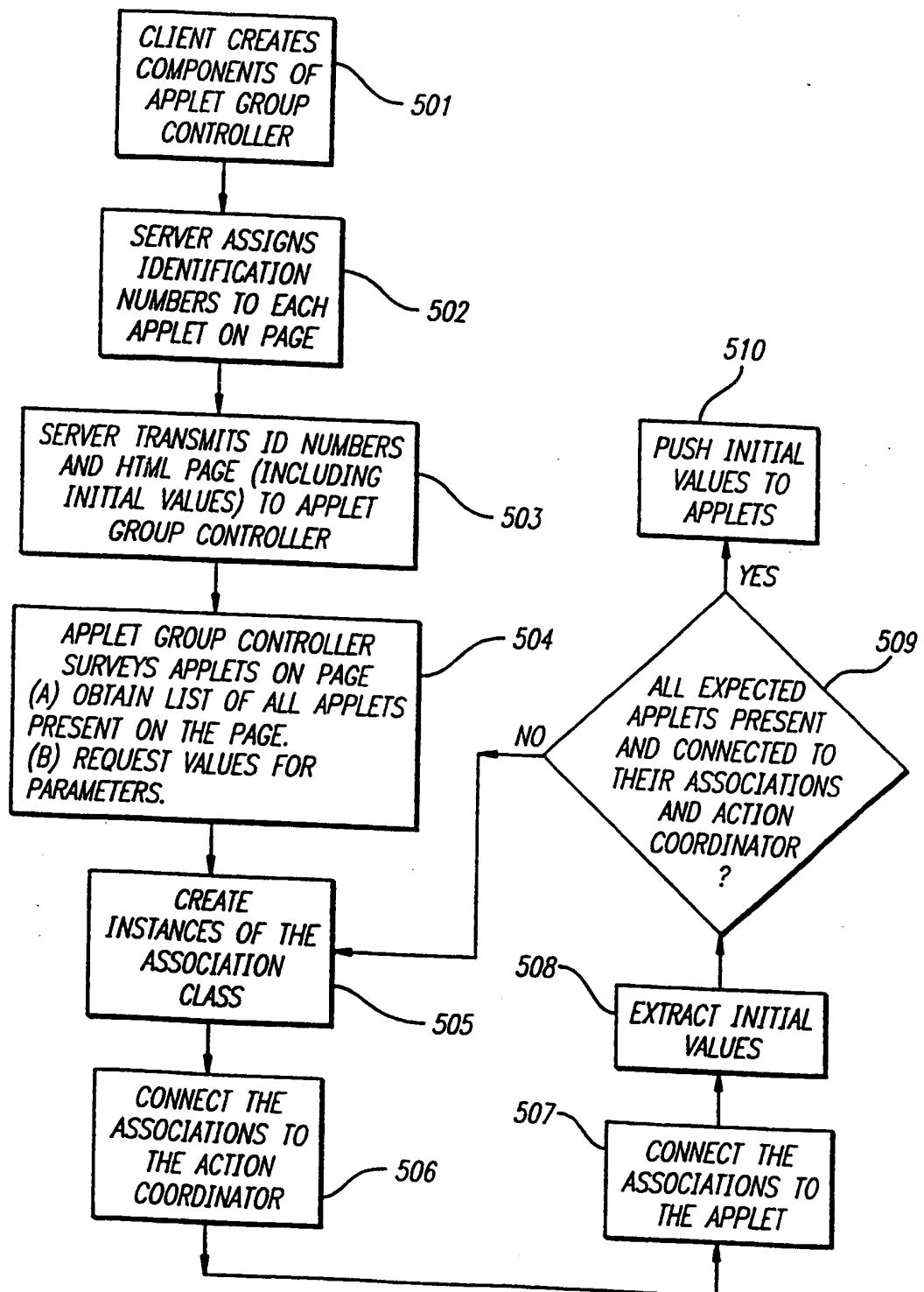
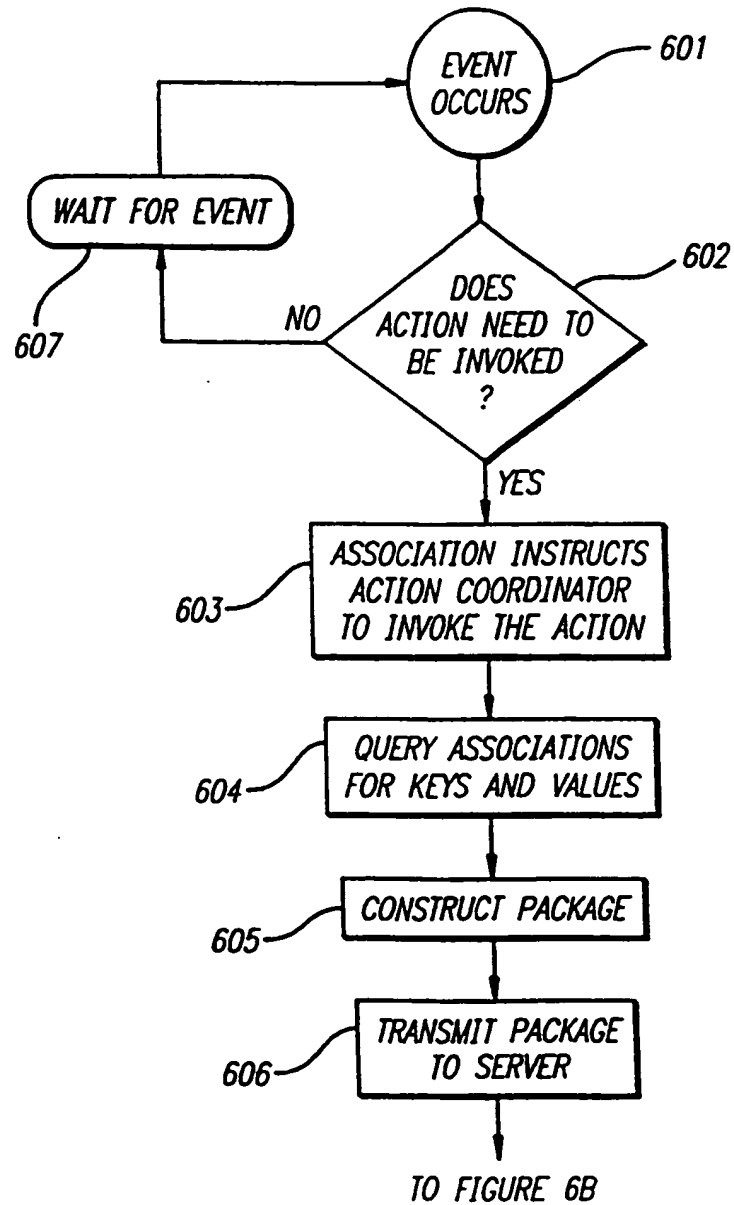


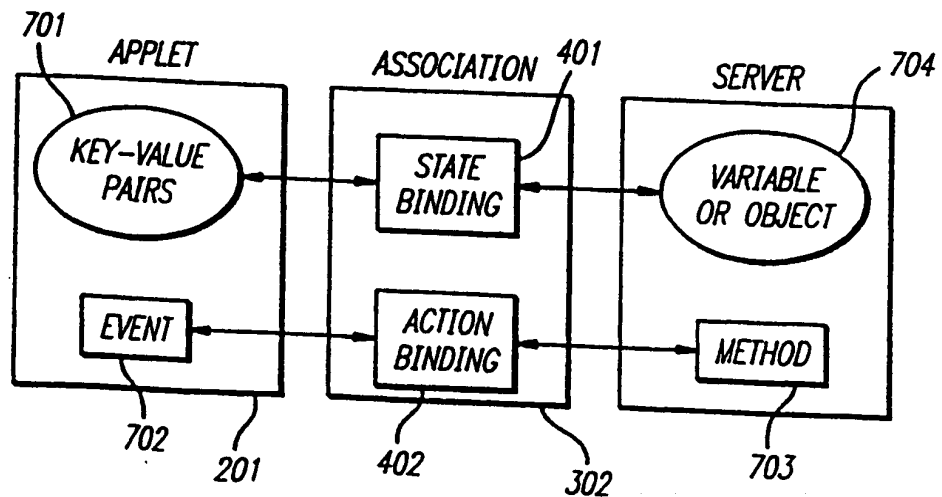
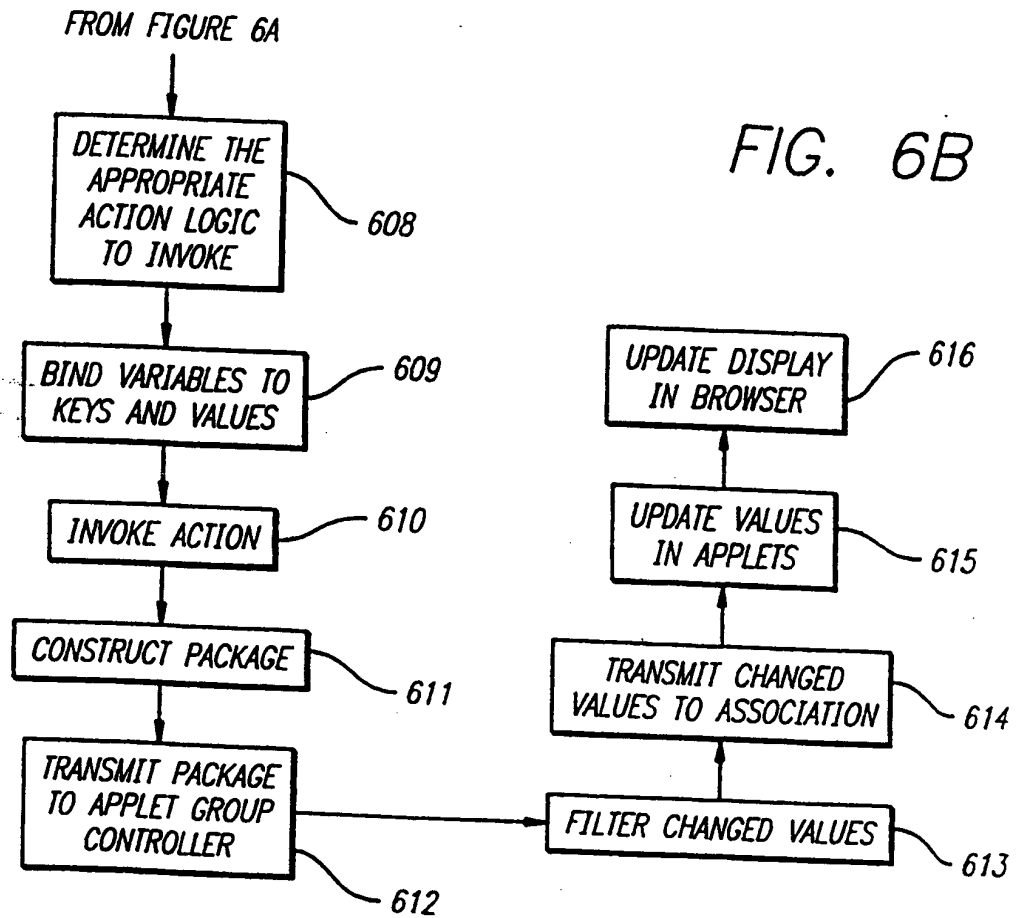
FIG. 5

4/5

FIG. 6A



5/5

**FIG. 7**

SUBSTITUTE SHEET (RULE 26)



# INTERNATIONAL SEARCH REPORT

In .ational Application No

PCT/US 98/06227

**A. CLASSIFICATION OF SUBJECT MATTER**  
IPC 6 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

IPC 6 H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>DUAN N N: "Distributed database access in a corporate environment using Java" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, page 1149-1156 XP004018216 see abstract see page 1149, left-hand column, paragraph 1 - page 1155, left-hand column, paragraph 2</p> <p style="text-align: center;">--- -/--</p>	1,17



Further documents are listed in the continuation of box C.



Patent family members are listed in annex.

\* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search

20 August 1998

Date of mailing of the international search report

08/09/1998

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Lievens, K

# INTERNATIONAL SEARCH REPORT

In International Application No  
PCT/US 98/06227

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>JACOBS S ET AL: "Filling HTML forms simultaneously: CoWeb - architecture and functionality"</p> <p>COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, page 1385-1395</p> <p>XP004018236</p> <p>see abstract</p> <p>see page 1385, left-hand column, paragraph 1 - page 1388, right-hand column, paragraph 1</p> <p>-----</p>	1,17

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER: \_\_\_\_\_**

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image  
problems checked, please do not report these problems to  
the IFW Image Problem Mailbox.**

---

